The Bell System Technical Journal
Safeguard Supplement
Printed in U.S.A.

SAFEGUARD Data-Processing System:

SAFEGUARD Data Reduction System

By E. S. HOOVER and R. A. JACOBY

(Manuscript received January 3, 1975)

In evaluating and certifying the SAFEGUARD ABM system, it was necessary to interrogate and analyze the massive volume of data generated during tests. The number of different reports, listings, and plots was so large and the variety so great that a flexible data reduction system had to be placed at the disposal of the user community. At the same time, the system had to be highly efficient and quickly available to be used at all. The SAFEGUARD Data Reduction System was designed to accomplish these objectives.

I. INTRODUCTION

Since any operational system must be tested, certified, and evaluated, provision of the means for doing so must be part of its design. The first step in certifying that a process is performing as specified is the recording of certain significant data during test runs. These data must be reduced and presented in a variety of ways. The Safeguard Data Reduction System (SDRS) fills this role by providing a flexible and highly efficient facility to serve the needs of the test teams.

The fundamental capabilities of SDRS had to be available when the testing began. The design for the real-time recording programs and the data reduction programs had to be coordinated, since the recording program serves as input to the reduction program. Short reduction program development schedules made necessary the use of certain preexisting designs and code, which had to be worked into the resulting system without compromising the other requirements. To accomplish this, a number of deliveries were planned, starting with the simplest and most basic features. Users who tried the first system were able to give SDRS designers useful feedback for future deliveries.

This paper discusses the experience gained in formulating requirements, organizing the program, developing the facility, and interacting with users.

II. REQUIREMENTS FOR THE SAFEGUARD DATA REDUCTION SYSTEM

To test, certify, and evaluate the behavior of a SAFEGUARD process, it is necessary to record data from memory during the real-time execution of the process. Debugging and integrating can conceivably be accomplished by stopping the process and taking post-mortem memory dumps, but this destroys the real-time sequence of events. Since the SAFEGUARD processes contain hundreds of thousands of lines of code, testing would be exceedingly cumbersome if it were necessary to stop the test to record data. Because calls to the recording subroutine are planned for and remain always in the code, a wealth of internal data can be recorded without disturbing the real-time behavior of the process. Recording occupies some CPU time but, aside from this effect, the process performs in the same way whether or not recording occurs. Real-time recording is essential for a practical testing program.

2.1 Requirements for recording

Thousands of events for which data might be taken occur during a test. The specific data needed depend on the purpose of the test. The CLC operating system[1] allows the applications process to record up to 100,000 32-bit words of data per second, as many as eight reels of tape during a 16-minute test.

Unless care is taken in recording design, even this large capacity can be exceeded. Designers tend to do more recording than is needed. This happens because the recording decision must be made long before testing begins. By recording almost everything, designers protect against overlooking some data items that may be wanted. As a result, a burden is placed on data reduction to select from a large mass of data only those items the user needs.

The data as recorded by the CLC operating system are organized into physical records of variable length. Each physical record contains a header and one or more logical records. The header preceding each logical record categorizes the data to follow. Records of one type would contain the most common and essential items, while records of another type might contain more voluminous data.

2.2 SDRS requirements

Consideration of the content, structure, and volume of the input and the expected use for the output dictate requirements for the data reduction system. Very large quantities of data are recorded from over 1000 different data sets.

Many data structures are implemented in the processes, but four typical ones were selected for processing by SDRS. Many more complicated structures, if properly recorded, can be handled by SDRS.

To be correctly interpreted by SDRS, the physical attributes (floating point, integer, etc.) of each item of recorded data must be defined. Since a majority of these items must also be defined for CENTRAN compilations, SDRS avoids possible incompatibilities by accessing the CENTRAN declarations.

To reduce data items not defined in CENTRAN and to allow quick response to patches in the CLC applications processes, SDRS also provides utilities to define and modify attributes. Experience with earlier data reduction systems, which provided only manual methods of data attribute definition, led to the requirement for both automated and manual methods.

There is a need to format the data so that they may be interpreted with ease. Some factors to be considered are discussed here.

Ease of interpretation of the recorded data requires that methods be supplied for selection of only the necessary subset of the data for presentation. This allows the user to generate exception reports and summaries rather than printing or plotting every data value.

Raw data may be recorded in one form, but they may be much more useful in another. Presentation in engineering units is often helpful. User-defined computations can be made by SDRS to facilitate evaluation of data.

In some cases, related data are scattered over a series of records and even over a series of tapes. The difficult task of correlation is performed by a file handler that can associate data for a user.

Some forms of presenting data are more useful than others. Since it is impossible to predict what particular listing or plotting form will best serve a given user, the users need the ability to format their own reports for presentation. Specifically, the users choose from among four basic ways to present data: formatted reports, tabular listings, line and point plots, and histograms. Users specify titles, subtitles, column headings, plotting axes, and scaling parameters for plots.

To gain user acceptance, SDRS (which is not a real-time facility) had to provide features that could satisfy quick turnaround time requirements with minimum effort.

Since most users developed their requirements as they began testing, SDRS had to provide users with plots and tabular listings which they could then easily modify to suit their later needs. SDRS provides users with a high-level command language in which to specify more complicated reduction requirements. Sufficient default conditions are supplied so that a simple set of user commands will result in a listing of all data items in a logical record.

One group of five users designed 737 tabular listings and reports in 12 months, averaging 12 per man-week. The elapsed time to get a simple tabular listing to work was about two days.

Because thousands of corrections had to be made to applications processes, there was a premium on quick turnaround time in processing the recording tapes. Since many testing teams submitted reduction runs simultaneously, it was essential that SDRS process a large volume of records efficiently.

It would be possible to reduce data on the CLC (this will be done during the post-installation and test phase). However, the CLC installations are limited in number, and time on the CLC is normally devoted to testing. Therefore, a decision was made early that data reduction should be done off-line on commercially available computers. Testing is performed for SAFEGUARD in several locations. In each of these locations, an off-line computer of the IBM 360/370 series is used for data reduction. The decision to use an off-line facility was correct. Time on the test machine is limited and is in much demand for the primary testing task.

A facility such as SDRS cannot be developed with all capabilities operational at the time an initial capability is needed by users. The designer can capitalize on this. If he designs a modular system and an open-ended user command language, he can first deliver a simple system. Feedback after the users have tried the first system can improve the design of later extensions. In fact, while users were presented with a proposal for SDRS and were invited to give their comments, most suggestions were obtained only after the first version of the system was working.

2.3 What was learned in setting requirements

The needs of users with a great many reduction requests to maintain were not foreseen. Since SDRS made it easy to request a large number of different reductions, the administration of requests required automation. One user group, supporting a single process integration, generated over 7000 SDRS statements. User groups usually solved this problem by developing their own administrative programs.

It was important to restrict users to one specific command language to give them the ability to turn out data reduction requests quickly. Some users, familiar with FORTRAN or PL/I, undoubtedly would have liked full control over program execution and the use of data types available in those languages. Restricting the number of data types supported increased the speed with which the system was developed.

III. SYSTEM ORGANIZATION

3.1 Design considerations

The development of a system organization and design philosophy for SDRS was influenced by a variety of factors. These included user

requirements, schedule constraints, experience of the designers, and successes and failures of earlier systems.

The designs of several previous data retrieval and analysis systems were analyzed to determine their applicability to the system requirements. This analysis uncovered serious shortcomings in most of these systems for the high data volume requirements of SAFEGUARD; however, several common strong points were noted in each.

A general solution to the data correlation problem was attempted with the Mission Data Reduction (MDR) system, which was developed for use in the Meck test system. The significant features of this system were a command language user interface, general data sorting capabilities, general data conversion capabilities, and data presentation capabilities in the form of reports, plots, and tabular listings. The major shortcomings of MDR were its complete dependence upon sorted data to produce any output as well as a requirement to convert all data before selecting the subset of interest. Both these characteristics introduced exorbitant overhead for sequential processing.

A more specific approach to the problem was used in the systems that were successors to MDR. These systems added a data attribute dictionary, an efficient sequential-file data extractor, and specific data correlation capabilities in the form of special-purpose subroutines to the basic capabilities of MDR. The major shortcomings of these systems were limited selectivity during the extraction phase, a requirement to convert a large percentage of the total data before selecting the subset of interest, and limited file generation capabilities that required many passes over the raw data to extract all data of interest. An additional shortcoming was a dependence upon manual maintenance methods for the data attribute dictionaries.

A common factor in each of these systems was uncovered: The designers had little or no control over the format of the data files that they were required to process. In general, the files were written without regard to the eventual processing and correlation requirements. With few exceptions, these characteristics of the data to be processed introduced a great deal of complexity and overhead into the systems.

Once it had been determined that none of the available systems would meet the requirements adequately, a design was proposed that would provide an initial capability with incremental growth potential. The user community required delivery of the first release within nine months and incremental releases at two-month intervals.

To meet these schedules, it was necessary to achieve a balance between the development of new programs and the adaptation of existing programs. The advantages of short development time offered by use of existing programs had to be weighed against their extendibility,

flexibility, and maintainability. In addition, the efficiency requirements for the data presentation capabilities had to be considered.

The requirements obtained from the user community were analyzed to determine possible commonality. An attempt was made to determine the general characteristics of the data that would be generated by the users. An estimate of the probable volume of data to be generated by these users was also made.

All users requested the same basic capabilities. These included printing reports in a variety of forms, correlating and sorting data on a variety of criteria, plotting data, and specifying the conditions under which this processing was to be done. The large number of special processing requests made it obvious that the development of a general-purpose facility was necessary.

3.2 Basic functional components

The system consists of four basic functional components as indicated in Fig. 1:

(*i*) The data attribute definition component defines the characteristics of data items by examining their CENTRAN declarations.

(*ii*) The sequential data base retrieval component provides data collection, selection, and presentation capabilities for sequentially organized data files.

(*iii*) The hierarchical data base generation component allows the relatively efficient creation of direct access data files.

(*iv*) The hierarchical data base retrieval component provides data collection and selection capabilities as well as sequential data base generation capabilities for direct access data files.

3.3 Lessons learned

The overall efficiency of SDRS is difficult to measure since the users of the system specify what the system must process. This introduces into the evaluation of SDRS performance such factors as user expertise, user knowledge of data characteristics, and user analysis of needs. Several design decisions were made to minimize the impact of these factors on system performance.

Provision of methods for the user to perform many data presentation operations on a single data retrieval pass was the primary characteristic of the design that provided efficient processing capabilities. This approach, although somewhat obvious for processing sequential data bases, is equally applicable to the processing of direct-access data bases. This is true because minimization of the number of times data are retrieved will minimize elapsed time and system overhead.

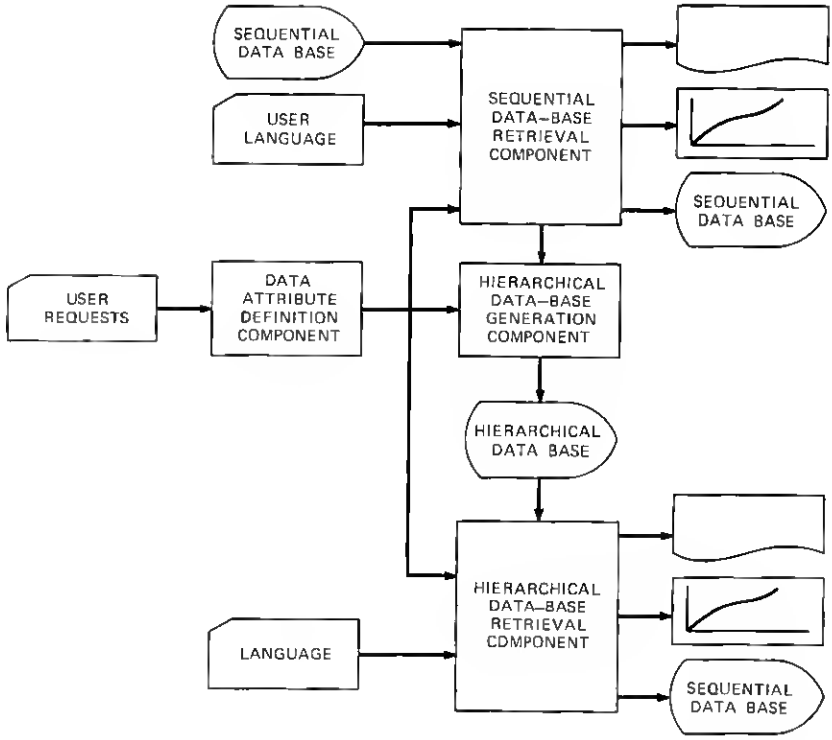

Fig. 1—Basic functional components.

Structured data recording makes it feasible to develop a simple efficient data-filtering algorithm. This algorithm enables SDRS to discard all records not requested by a user by interrogating fields in the header and ignoring all data in the record.

Limiting the number of data structures supported makes it feasible to design algorithms that extract and convert a minimum amount of data. Minimizing the number of data conversions was especially critical in the sequential data base retrieval module because of the large volume of data. In this module, data conversion is delayed until after all user conditions had been satisfied.

Assembly language was used in coding those critical paths of the system that would process large volumes of data. The extra time required to develop these programs was offset by the increased efficiency derived from this approach.

IV. DEVELOPMENT CONSIDERATIONS

The development of SDRS and the delivery of the system to the user community with capabilities consistent with the requirements were

accomplished on short schedules. Several development procedures and techniques were used by the design group that may be applicable to other development efforts faced with similar problems.

The critical need of many users for a simple data-printing capability was the basis for the design of the first release of SDRS. This release consisted of the basic versions of the data attribute definition component and the sequential data base retrieval component.

Although simple from a user capability point of view, this initial release of the system was designed to be extendible. Emphasis was placed on development of a design that would allow inclusion of additional processing capabilities without major perturbations.

The development of an outline for further functional capabilities was begun in parallel with the development of the initial system. This outline served as a vehicle for planning capability development and delivery.

Formal design specifications for each system component were not written. However, detailed interface specifications were developed. This made it possible for individual routines to be designed in parallel.

The development of SDRS on schedule would not have been possible without the use of a time-sharing system. Although time sharing is relatively expensive, development times can be minimized when rapid correction of troubles and extensive testing are required.

The size of the system required that extensive testing be done to verify system performance. Although testing is possible in a batch environment, the effectiveness of the system test team was greatly enhanced by the availability of immediate test results and on-line debugging capabilities.

V. USER INTERACTION

Before SDRS was designed, users were asked to submit their requirements. The SDRS design group then proposed to users an initial set of requirements. Only after the initial release was meaningful user feedback received. Whenever possible, suggested improvements were incorporated into subsequent versions.

A system as complicated as SDRS requires user education. Two methods were used for this purpose: A user's manual was written and counselors were provided. The role of the counselors was to teach correct and efficient SDRS use and to collect feedback for improvements.

The final service to users is proper test and maintenance of the system. Users were not asked to be guinea pigs. They were allowed to try a new SDRS only after a complete set of tests were run. During two years of use, only 81 troubles were encountered in 105,000 lines

of source. Because of the error messages and the modularity of the system, it was easy to identify and fix problems.

Total effort expended in user services has been 18 percent of the manpower of the SDRS group. This is considered a minimum effective support level.

VI. CONCLUSION

The primary lesson learned from the development of SDRS is that user data base design is critical. Recording and reduction efficiency is achieved by designing data bases to minimize the requirement for further correlation and restructuring.

The real achievement of SDRS lies in simultaneously accomplishing the objectives of flexibility and efficiency. Many systems attain one goal or the other; SDRS attempted to do both. Two design decisions contributed to the success of this effort. First, recorded data not wanted by the user are ignored by the system. Second, once data are retrieved, they are processed in as many ways as needed.

REFERENCE

1. J. P. Haggerty, "SAFEGUARD Data-Processing System: Central Logic and Control Operating System," B.S.T.J., this issue, pp. S89–S99.